State of the art metrics for aspect oriented programming

Mazen Ismaeel Ghareb, and Gary Allen

Citation: AIP Conference Proceedings **1952**, 020107 (2018); doi: 10.1063/1.5032069
View online: https://doi.org/10.1063/1.5032069
View Table of Contents: http://aip.scitation.org/toc/apc/1952/1
Published by the American Institute of Physics

State Of The Art Metrics For Aspect Oriented Programming

Mazen Ismaeel Ghareb [1], Dr. Gary Allen[2]

[1]*Computer Science Department., College of Science and Technology, University of Human Development Sulaimani, Iraq*

[2] *Department of Informatics, School of Computing and Engineering, University of Huddersfield, Huddersfield, England*

mazen.ismaeel@uhd.edu.iq

g.allen@hud.ac.uk

Abstract. The quality evaluation of software, e.g., defect measurement, gains significance with higher use of software applications. Metric measurements are considered as the primary indicator of imperfection prediction and software maintenance in various empirical studies of software products. However, there is no agreement on which metrics are compelling quality indicators for novel development approaches such as Aspect Oriented Programming (AOP). AOP intends to enhance programming quality, by providing new and novel constructs for the development of systems, for example, point cuts, advice and inter-type relationships. Hence, it is not evident if quality pointers for AOP can be derived from direct expansions of traditional OO measurements. Then again, investigations of AOP do regularly depend on established coupling measurements. Notwithstanding the late reception of AOP in empirical studies, coupling measurements have been adopted as useful markers of flaw inclination in this context. In this paper we will investigate the state of the art metrics for measurement of Aspect Oriented systems development.

Keywords: Aspect Oriented Programming, OOP Metrics, AOP Metrics, Line of Code, Hybrid Aspect Oriented Development, AODL (Aspect Oriented Design Language)

INTRODUCTION

Static measurements are used to evaluate the quality of computer code. Cohesion, for example, is the degree to which components of a module work together with each other. Currently there are not many measurements for AO frameworks [6], for example, cohesion, and coupling, separation of concerns, size and so forth. Cohesion is one of the essential quality properties for AO frameworks. Coupling measures the level of interaction between modules. Low coupling and high cohesion are considered essential for a good design. In [4], W. Cazzola inferred one structure in view of element dynamic metrics and run time execution of software reports from several research papers [5-8], and concentrates on element estimations dismissing static, traditional measurements. The importance of software measurement has been increasingly recognized by Object Oriented (OO) software engineers as the metrics have been proven to be pointers of vital quality properties, for example, the fault-proneness of the final system [7] [25]. In this manner, the quality pointers for AOP can be gotten from direct expansions of traditional OO measurements. In any case, observational investigations of AOP do frequently depend on established coupling measurements.

LITERATURE REVIEW

The experimental studies are required be conducted to evaluate the costs and the advantages offered by the AOP when compared to the more traditional Object-Oriented (OO) approach. Such studies should be cover code understandability, resolvability, measured quality and testability. Additionally, selected AOP constructs could be differentiated empirically, in order to distinguish good/bad AOP practices, and to identify a list of common and reusable AOP designs. The initial phase toward this step is the definition of a set of measurements to quantitatively survey the impacts of the AOP approaches, i.e., a set of OO Metrics and AOP Metrics.

International Conference on Electrical, Electronics, Materials and Applied Science
AIP Conf. Proc. 1952, 020107-1–020107-10; https://doi.org/10.1063/1.5032069
Published by AIP Publishing. 978-0-7354-1647-5/$30.00

Object Oriented Metrics

The insufficiency of the measurements being used with procedural code (complexity, size, and so forth.), when connected to OO frameworks, prompted the creating of new measurement suites representing the particular elements of OO programming. The one that is most widely received and referenced is that by Chidamber and Kemerer [8]. Some metrics utilized as a part of Chidamber and Kemerer's suite can be effectively adjusted to AOP programming, by binding together classes and Aspects, and also methods and advices. Aspect presentations and static crosscutting require minor adjustments [25].

Nonetheless, many works have been done regarding of coupling metrics of AOP, requiring particular estimations. For instance, a method execution is captured by an aspect pointcut, triggering the execution of an aspect, makes the caught method combined with the advice, in that its conduct is potentially changed by the advice. In the reverse direction, the aspect is influencing the module containing the captured operation, along these lines it relies upon its interior properties (method names, control stream, and so forth.)

Aspect Oriented Metrics

According to [9] many measurements apply both for classes and aspects. In the following discussion the term module will be utilized to refer to both of these two modularization units. So similarly, the term operation subsumes class methods and aspects advices. So here are some candidate aspect metrics.

DIT (Depth of Inheritance Tree): Length of the longest way from an offered module to the aspect hierarchy root. The more profound an aspect is in the hierarchy, the number of operations it might inherit, hence making it more complex to understand and change [10].

Size and complexity. NOM (Number of Methods) and NOA (Number of Attributes) are used to measure the size of the class in terms of method and attributes. WMC (Weight Method Complexity) and CC (Class Complexity) are connected to measure classes and are used to measure total complexity through calculating the total number of functions/methods in different ways. Since classes are proposed to be designed as succinctly as possible, these measurements are required to be low in their qualities [11].

Cohesion: Cohesion is measured with four class-level measurements, which are calculated in various approaches to reflect the collaborations between part function/methods. The four levels are LCOM (Lack of Cohesion in Method), TCC (Tight Class Cohesion), LCC (Loss Class Cohesions) and ICH (Information Based Cohesion) [12].

Reusability: RR (Reuse Ratio) and SR (Specification Ratio) are both framework level reusability measurements. They are calculated as the ratios of subclasses to all classes and to super classes, separately. Since classes are relied upon to be profoundly reused, extensive reusability metric qualities are desired [13].

Polymorphism: NMO (Number of Methods Overridden by the Class) and PF (Polymorphism Factor) are polymorphism measurements at various levels. To be particular, NMO is a class-level metric, which measures the number of methods overridden by a singular subclass, while PF is a framework level metric, which measures the degree of method overriding in the entire system [14].

Encapsulation: MHF (Method Hiding Factor) and AHF (Attribute Hiding Factor) are indicators to show how well methods and attributes are hidden inside classes. These measurements are measured at the framework level [15].

Coupling: Five measurements are used to assess class coupling from alternate points of view. The CF (Coupling Factor) metric is used to assess the coupling of all classes at the framework level. By examination, the other four measurements measure coupling at class level. Among these measurements, RFC (Response For the Class) and MPC (Message Passing Coupling) are utilized to survey technique coupling, DAC (Data Abstract Coupling) encapsulates information coupling amongst classes, and CBO (Coupling Between Objects) indicates coupling between class occurrences [16].

METHODS AND MATERIALS

The methodology of this research is to survey all existing research into Aspect Oriented Programming metrics. These metrics are used for evaluating the quality of AOP system. This research identifies metrics that have been discovered by many researchers in recent years. The metrics include Weighted Operations in Module (WOM), Depth in Inheritance Tree (DIT), Crosscutting Degree of an Aspect (CDA), Coupling between Modules (CBM) and Lack of Cohesion in Operations (LCO). Many researchers have shown different method for identifies these metrics and still there is not unique method for proof the evaluation of the metrics.

According to [17] they have been used visualization techniques for indicating the measurements of Aspects. The Work extends the perception system VERSO.The Representation of Aspect Metrics were the elements that they chose were the 3D box, the cylinder and the pyramid which are basic and have a few fascinating attributes, for example, shading, bends and height. The Selected System were The RASim framework is an intelligent graphical delicate product test system intended to teach tool for robotics. JHotDraw extend was developed by Erich Gamma and Thomas Eggenschwiler. It is a Java GUI system for technical usage and graphic design. Paper [18] Concern about examine AOP and OO Versions of OpenBravoPOS and Jasper reports. The versions of AOP shown changes in all the basic complexity measurements. Including more it demonstrates a general improvement in the coupling. The results of the contextual investigation are empowering in that it shows the reasons of the aspect oriented approach in enhancing maintainability in general.[19] supposed an analytical evaluation, and experimental information from ten open source ventures, deciding the estimation of six measurements for AOP (Crosscutting Degree of an Aspect (CDA), and Coupling on Aspect Execution (CAE), Number of Child (NOC), Lines of Code (LOC), Depth of Inheritance Tree (DIT) and Weighted Operations in Module (WOM)). This work gives an arrangement of contributions to the utilization of measurements for Aspect Oriented programming. The systematic assessment of the chose measurements against built up criteria for legitimacy demonstrated that the Aspect Oriented adapted metrics, by and large, fulfill the criteria initially fulfilled by the object-oriented metrics, which implies that they can be utilized to survey perspective arranged programming and give practical comparable results.[12] Approach is the principal proposition in aspect cohesion metric. It depends on a dependency model for AOP that consists of a gathering of dependency diagrams. They guarantee that cohesion is characterized as the level of relatedness amongst attributes and modules. So for that more investigation and utilizations of attributes in setting to applications required. The authors likewise talked about the numerical properties of these coupling measures, in which he demonstrated that these measures fulfill properties that a decent coupling measure ought to have.[20] proposed metric for measuring cohesion and it is contrasted and existing cohesion measurements for the Aspect Oriented system. They characterized cohesion on the premise of dependency analysis. In their work they considered two criteria's for measurement Modules - Modules and Modules - Data association criteria.[21] Examined about the Aspect Oriented framework and its advantages position over module-oriented and object oriented framework. In this paper real concentrate is given in different cohesion measurements accessible for heritage framework and also for aspect-oriented framework. [22] In this work the authors characterized new terminologies for representing components of AO systems. In their structure, they considered two AOP languages, AspectJ and CaesarJ. They have broadened their structure from Briand's systems and endeavored to expand the utilization of the terminology and coupling criteria initially characterized in Briand's structures.[23] The results of this study showed that coupling measurements, which are not directives of Object Oriented measurements, had a superior to be unrivaled markers of fault-proneness. Results demonstrate that Base-Aspect Coupling (BAC) and Crosscutting Degree of an Aspect (CDA), while the two measurements that showed the most grounded relationships with faults. They outperformed an assortment of well known measurements generally utilized as a part of AOP observational investigations, including cohesion, size and other coupling measurements.[26] This research is studying the effects of coupling metrics of AOP on maintainability of software. The use 14 journals from different database to focus on the coupling of AOP regards the maintainability issues. The WOM and DIT is improved even between three different AOP languages while the coupling between modules is variable but generally improved over OOP coupling Metrics. [27] This research has been worked on JHotdraw software which developed with Aspect. The AspectJ objects represent 10% of the all source code. The final results show improvement of WOM, LOC, and CBM. The drawbacks were in DIT and LOC. [28] This paper has accomplished a comparable study on implementation of design patterns of GoF using OOP and AOP, in order to test the complexity of the two methods of implantations. The results showed that there is not improvement in all design pattern, it depends on the current problem type some of them improved other not.[29] This paper has been tested aspect oriented programming using UML profile techniques. They used same OOP metrics that suitable for AOP. The outcomes showed that there were improvements in metrics when using AOP development. [30] This paper has proposed a metric for AOP. They have implemented an online shop application for both OOP and AOP. They try difference scenarios for testing the complexity and cohesion and coupling. The results showed high cohesion and low coupling in their AOP implementations. Adding more AOP enhance the maintainability and reusability in their implementations. [31] This research studies the software which has %1 of implantations of AOP. They investigate AJHotDraw. The outcomes showed that some metrics have been improved such WOM, LCO, CBM, While there are drawbacks in the DIT. [32] This paper proposed a framework for AOP metrics. The experiment showed the implementation of a Portal ware system into different versions, both OOSD and AOSD. They proposed some new AOP Metrics of cohesion and coupling of Aspect, Also showed some drawbacks of AOP implementation regarding the OOP as it is shown in the results section. [33] This Research has a theoretical

evaluation of AOP metrics of 14 papers. They have proposed new metrics of cohesion and coupling and separation of concerns which improve the evaluation of AOP software quality. Adding to that the WOM was increased in AOP implementation in most cases. [34] This paper investigates the metrics of AOP by an empirical experiment. They have developed a web information system using OOP and AOP. The comparison showed that AOP have slightly improved over OOP. The experiments repeated after maintaining it for both versions OOP and AOP. AOP still has improvement on OOP. [35] The research proposed an evaluation artifact model for evaluation reusability of AOP. The results showed that AOP implementation has slightly improved on OOP implementations and LOC, DIT and WOM metrics. [36] This research has been proven that AOP implementation significant effect on design quality on OOP. They have evaluated these metrics DIT, WOM, CBM and LOC. The experiments, implementation has done by implementing design pattern observer pattern in both OOP and AOP versions. [37] This study has shown an evaluation of AOP metrics. They have analysis mobile middleware system COS. They have tested several factors of maintainability, reusability and sub-characteristics and found several AOP metrics. The outcomes revealed improvement in AOP implementations regarding these above factors. Moreover improve in WOM, DIT, and CBM. [38] This paper concern in evaluating AOP metrics of Coupling and Cohesion. Their experiment was implemented 23 design patterns of GoF. The implementations were into versions OOP and AOP. The outcomes show that AOP coupling which means molding is better that OOP.[39] The research concerned with comparing between OOP and AOP implementation. The AOP improves both coupling and cohesion. They have used time traffic simulator for evaluation. [40] This study has been evaluated AOP metrics. They have used Health Watcher HW. They found that modularity is improved by AOP and the cohesion by 17%.[41] This Study have develop middle ware of Cisco with AspectJ tool. The results showed 50% of improving in modularity and quality regarding other development tools. Moreover it also reduce complexity of the system when usinf AOP. [42] This study work on implementing scattered of kernel code the operation system using AOP. The focused was on four factors modularity, changeability, configurability's and reduces redundancy. All these factors have shown improvement regards other implementation techniques. [43] This study focused on finding AO effects on software metrics. They have propose UML extension of OOP combines with AOP paradigm. They have shown improving in WMC, DIT , and CBM but not big improvement in cohesion (LOC) between classes and aspects. [44] This paper focused of proposing new method for measuring AOP Metrics . The have using Method Graph Dependence (MGD) for testing the complexity of the system. They have tried many modeling for measures these parameter. The outcomes was the modularity and complexity improved when using AOP by applying their method, However more experiments need to done on cohesion and WOM ,DIT ,CDA and other AOP metrics. [45] This paper was a comparison study between mobile application development using two different techniques, java and AspectJ. The results show that AspectJ implementations are better than java implementation regarding complexity and modularity. This means WOM, DIT and coupling between classes and aspect. [46] This study have worked on measuring the complexity and modularity of mobile agent application for both implementation Java and AspectJ . They proposed an complexity metrics for AOP using entropy complexity method. The experiments showed improvement in AspectJ development in DIT, WOM, CBM, but not in cohesions. Adding to that improve the usability and reduce complexity. [47] This research is a review paper on coupling metric measurement techniques on AOP. The majority of the studies have shown a massive impact on software quality metrics such as WOM, DIT and coupling when using AOP development techniques. [48] This evaluation study of AOP metrics among many research papers. The several studies showed that AOP is not necessary to reduce the complexity by it is own. Regarding the coupling and cohesion it reveal good quality of software when using aspect development techniques. Several studies shows improve modularity when using crosscutting concerns in software developments. [49] This Research investigated the effect of Aspect development on software qualities. The results was in two diffrent manner on class level there was no positive impact, while on package level aspect shows improvement in software design quality. Their methodology was Xtreame programming methodology of 30 developers. They have developed web system for submission and review. The applied three case studies two of them using OOP and one AOP.The measurements metrics were LOC, WOM, DIT, CBM and LCO. Generally it needs more experiments to prove these controversial results. [50] This paper have been examine the effect AOP development on software quality metrics. The case study was programming Observe pattern, AJHotdraw with AspectJ and compare it with Java language. Generally the AOP showed improving in modularity and cohesion. however there is lack of measure aspect crosscutting effect on system components, it will need more investigations. [51] This study was reveal of effect aspect development on software development quality and software metrics. The development of 14 design patterns is shown improvement of maintainability, complexity and modularity of these patterns. Adding more the metrics of using AspectJ development shown significant improvement compare of OOP programming, because the functionality and implementations are the same for design patterns coding. [52] This research will focus on development using

AspectJ the case study was development of 23 design patterns using aspect. The new way of crosscutting implementation improved modularity, readabilities, comprehensive understanding of inheritance and improve modularity. These experiments reveals it needs to investigate more on AOP metrics such as points cuts effects, Aspects and advice. [53] This Research has a deep investigation on crosscutting concern metrics for AOP. The worked on finding the relationship between advice and class have been tested. They have measured the effect of piece of code between the classes and ApsectJ components. The studies have worked on crosscutting, cohesion and coupling metrics. Four case studies of source code have been tested ATS, Prevailed, AJHotdraw and FACET. Generally the AOP shows improvements in development process. [54] This research concerned with evaluation of AOSD metrics regarding the effect on software quality. The study has examined the crosscutting concern metrics and has calculated it manually. Regarding the coupling metrics it has found that several coupling types between aspect, OOP and AOP metrics which were not found in previous studies. These types of coupling will improve the quality of software development, adding to that cohesion metrics have the same issues , for instance lack of cohesion for method if you measure it will be the same for advice. Size metrics is critical because if we have more code it affect on usability's and maintainability, for example. If you avoid duplicate for Aspect point cuts is very difficult to maintain. [55] This paper has studied three case studies for proven cohesion metrics of AOP on software quality. Three case studies have been examined design patterns, AJHotDraw and UML tool. The results show that cohesion metrics improved in software quality for all the three case studies. However there is varying results because the different types of programming. [56] This research is worked on several case studies to prove cross cutting concern that improve modularity. They have tested cross cutting concern by working on experiment using mathematical equation to make this approach. The results shows that 53% of case studies show improvement on modularity of the cross cutting concerns. [57] This research has been proposed and developed a system for measuring coupling and cohesion of AOP. The applications have been developed using .Net framework. The experiment has tested coupling, cohesion, line of code, cross cutting concerns and depth of inherence of AOP components. The results showed significant improvement of AOP development over classical OOP development. The proposal has been tested with several scenario case studies. [58] This research has been focus on study measuring static and dynamic coupling of OOP. They have been developed a system using Ncrunch using C#. The coupling and cohesion is an internal component for evaluated the software qualities. The static metrics measure the expected relations between components which was a classic way of measuring, while dynamic metrics measure the actual relation between components on runtime. AOP used to measure dynamic metrics for coupling and showed improved in the results. [59] This work has been worked on 75 similar subject in development process of AOP and OOP. They have proposed QAM-AOP model for this study. The studies have proven improvement of AOP on quality characteristic and readability which means reusability and maintainability. The result shows the improvement the coupling, cohesion, CBM and size when using AOP software development. [60] The study concerned with testing scalability of AOP in GoF design pattern. It has a empirical study of testing 62 design pattern compositions and compared with OOP. They have been tested most of AOP metrics DIT, WOM CDA, CBM and LCO. Most of the results show improvement in modularity of design and good scalability of AOP implementation over OOP. [61] The Study has been tested the reliability of aspect oriented programming using new method of artificial neural network. Reliability it means the quality of the software. The new method was using Multilayer Preceptor Artificial Neural Network (MLPANN). The experiment results have shown improvement in quality of AOP in several metrics (DIT, WOM, CBO, CDA, CBM and LOC) which also lead to improve the reliability of. [62] The research has worked on developing a 23 GoF design pattern with Aspect Oriented programming. Most of the results showed improvement in development process of cross concerns except 4 design pattern. Moreover it has improved the coupling, cohesion and size of selected design patterns, However more experience need to be done in order to prove all the scenarios. The recommendations of this study have shown good improvement in software quality with AOP. [63] This work have been tested a development of e-learning website using AspectJ. It have been also compare to OOP development. The AOP development shows improvement in maintainability and modularity with cross cutting concerns. Adding more they used design pattern such as Builder, structure Composite and Facade , behavioral chain of Responsibility and Strategy patterns. The aspect developments have shown significant improvement in software quality metrics such as line of code, modularity and cohesion in building the web tool using design pattern compare to OOP implementations. [64] This research has been worked on quantitative experiments of testing Aspect oriented programming with six design patterns. The study compares their results with OOP implementation of these patterns (Observer, state and abstract Factory, Prototype and Strategy, Mediator and Prototype). The suite of metrics they have chosen for measuring separation of concerns, coupling, cohesion and line of code in their experiment. The overall results have shown improvement of crosscutting concerns and cohesion in AOP, while there are high coupling and line of code in some of implementation. [65] This paper has been worked on developing a software

tool for measuring Aspect Oriented metrics using Query based analysis. Software developed using consist of Xquery, java/aspectJ and XML. Five open source projects developed with AspectJ have been examined (AJHSQLDB, AJHotDraw, Contract4J, DJProf and AspectJ Exception Framework (AJEFW)). Several metrics have been used coupling, cohesion, depth of inheritance, weight of attribute, cross cutting coupling and weight operation per components. The overall all results have shown a significant outcome achieves when using Aspect Oriented development. [66] This study have been examine the effect the cross cutting concerns for Multiple Agent System (MAS).The empirical evaluation showed significant improvement in cross cutting concerns , low coupling, fewer line of code, lower cohesion and improvement in modularity of the too. The MAS is commonly developed with OO object oriented abstraction techniques and OO design techniques. So there is many OOP developments have been done for MAS. The comparison study have been done in this research which have been showed a significant improvement of development MAS with Aspect.

RESULTS

The results of analyzing theses paper can be showed in table below each paper contribution in Aspect Measurements metrics.

Source	WOM	DIT	CDA	CBM	LCO	Measurements Methods
[17]	-	-	-	-	+	Visualizations using 3D Objects
[18]	+	+	+	+	+	AOP and OO Versions of OpenBravoPOS and Jasper reports.
[19]	+	+	+	+	N.A	10 open source project (AspectJ Hot Draw, aTrack, Glassbox,etc..)
[12]	+	N.A	N.A	N.A	+	Mathematically proven with graph depended method
[20]	N.A.	N.A	N.A	+	+	3 case study GOF Design Patterns, Zhao and Xu Source code and Gregor Kiczales
[21]	-	-	N.A	+	N.A	A comparative study of several paper works on Aspect Orineted Metrics
[22]	N.A	N.A	N.A	+	N.A	A proposed new framework for Aspect Oriented Coupling Measurement
[23]	+	+	+	+	N.A	Case study open source iBATIS .
[26]	+	+	N.A	+	N.A	14 journal paper testing coupling in three AOP languages.
[27]	+	-	N.A	+	-	Implementing JHotdraw tool.
[28]	-	-	N.A	-	+	Implementation Design pattern GoF for OOP and AspectJ.
[29]	+	+	N.A	+	+	Using UML Profile for testing AOP metrics.
[30]	-	+	+	+	-	Design and implementation of online shopping application.
[31]	+	-	N.A	+	+	Implementing JHotdraw tool.
[32]	-	-	+	+	-	The development of two versions of the Portalware system based on both OOSD and AOSD.
[33]	N.A	N.A	+	+	+	Theoretical comparison between 14 AOP metrics papers.
[34]	+	+	+	+	+	Developing a Web information system in both OOP and AOP.
[35]	+	+	N.A	N.A	N.A	The Evaluation artifact model has been proposed for testing reusability of AOP.
[36]	+	+	N.A	+	+	Implementing observer design pattern for both versions OOP and AOP.
[37]	+	+	N.A	+	-	They have analysis mobile middleware system COS
[38]	+	+	N.A	+	N.A	Implementation of GoF for AOP and OOP versions.
[39]	+	-	N.A	+	+	Testing using Time Traffic Simulator
[40]	+	-	N.A	+	+	Testing Health Watcher HW tool
[41]	+	-	N.A	+	+	Middle ware system Cisco ONS
[42]	+	+	N.A	+	N.A	Implementing Operating system
[43]	+	+	N.A	+	-	Propose UML extension for AOP metrics
[44]	+	+	N.A	+	N.A	Method Dependence Graph (MDG)
[45]	+	+	N.	+	N	Mobile Agent application implemented in AspectJ/HyberJ

			A		.A	
[46]	+	+	N.A	+	-	Entropy complixity measuer based on Fortran and Conbol language.
[47]	+	+	N.A	+	N.A	Review paper on coupling metrics on AOP.
[48]	+	+	N.A	+	+	Evaluation of AOP metrics among several studies.
[49]	-	-	N.A	-	-	Xtream Programming methodology for develop web based retrival system.
[50]	+	+	N.A	+	+	Implementin AJHotdraw with Observer Design pattern using OOP and AOP implementation.
[51]	+	+	+	+	+	Impmenting 14 design patterns using AOP techniques.
[52]	+	+	+	+	+	Implementing 23 design pattern using AspectJ programming.
[53]	N.A	N.A	+	+	+	Implement 4 source of code ATS, Prevailed, AJHotdraw and FACET
[54]	-	-	-	+	+	Evaluate AOSD metrics by analysis several other AOP metrics case studies.
[55]	N.A	N.A	N.A	N.A	+	Three case studies design patterns , AJHotdraw and UML design.
[56]	N.A	+	+	N.A	+	Mathematical evaluation for crosscutting concerns.
[57]	+	+	+	+	+	Proposing system using .Net framework for measuring the coupling and cohesion metrics in AOP.
[58]	+	+	N.A	+	+	C# development using Ncruch. Using AOP techniques for measure dynamic coupling and cohesion.
[59]	+	+	+	+	+	Comparative study between AOP and OOP development of 75 subjects with same background. Proposed QAM-AOP model.
[60]	+	+	+	+	+	Proposed a system for evaluation AOP metrics by implementing 62 composion of design pattern both in OOP and AOP.
[61]	+	+	+	+	+	The new method was using Multilayer Preceptor Artificial Neural Network (MLPANN).
[62]	-	+	+	+	+	This study has worked on measure aspect oriented development cross cutting comparing with OOP implementation. The have developed design of pattern in their study.
[63]	+	+	+	+	+	Development web tool for learning using AspectJ language. The study has compared it with OOP implementations.
[64]	-	+	+	-	+	Quantitative study of comparing OOP and AOP implementation of 6 design patterns.
[65]	+	+	+	+	+	Developing a metrics tool for measure AOP metrics using java/aspectJ, XQuery and XML.
[66]	+	+	+	+	-	Empirical study of testing separation of concern of multiple agent system using OOP, AOP and design pattern implementations.

TABLE 1 papers summarizations for AOP metrics contribution

DISCUSSION

The Results show that most common metrics of OOP can be applied on AOP. 23 papers with the AOP metrics subject have been chosen. Most of the AOP metrics show the improvement over OOP in their software development stages. Several source code and methods have been used, all these analysis leads to that majority of metrics that we have chosen shown improvement in AOP techniques.

CONCLUSION

In this paper, we have chosen 6 AOP metrics which can be applied on OOP also. The common metrics can be compared easily. Most of the paper showed an improvement of AOP implementation of OOP. However, some authors argue it depend on the language of implementation and software design problem. More than 25 source code of 32 papers has been evaluated theoretically and the majority proved that the AOP implementation Improve modularity, coupling, cohesion, maintainability and reusability. Most of these metrics were developed statistical or mathematically proven. Our intention is to build a framework model to evaluate AOP metrics dynamically without needs to terminate the system.

ACKNOWLEDGMENTS

We would like to thanks University of Human Development for it is usual support for research contribution.

REFERENCES

1. L. Cheikhi, R. E. Al-Qutaish, A. Idri, and A. Sellami, "Chidamber and kemerer object-oriented measures: Analysis of their design from the metrology perspective," International Journal of Software Engineering \& Its Applications, vol. 8, no. 2, 2014.
2. A. Kaur, S. Singh, K. Kahlon, and P. S. Sandhu, "Empirical Analysis of CK \& MOOD Metric Suit," International Journal of Innovation, Management and Technology, vol. 1, no. 5, p. 447, 2010.
3. N. Fenton and J. Bieman, Software metrics: a rigorous and practical approach. CRC Press, 2014.
4. W. Cazzola, A. Marchetto, and F. B. Kessler, "AOP-HiddenMetrics: Separation, Extensibility and Adaptability in SW Measurement.," Journal of Object Technology, vol. 7, no. 2, pp. 53–68, 2008.
5. E. Arisholm, L. C. Briand, and A. Foyen, "Dynamic coupling measurement for object-oriented software," IEEE Transactions on software engineering, vol. 30, no. 8, pp. 491–506, 2004.
6. M. Ceccato and P. Tonella, "Measuring the effects of software aspectization," in 1st Workshop on Aspect Reverse Engineering, 2004, vol. 12.
7. A. Mitchell and J. F. Power, "Using object-level run-time metrics to study coupling between objects," in Proceedings of the 2005 ACM symposium on Applied computing, 2005, pp. 1456–1462.
8. Chidamber, S.R. and Kemerer, C.F., 1994. A metrics suite for object oriented design. *IEEE Transactions on software engineering*, *20*(6), pp.476-493.
9. A. Kaur, S. Singh, K. Kahlon, and P. S. Sandhu, "Empirical Analysis of CK \& MOOD Metric Suit," International Journal of Innovation, Management and Technology, vol. 1, no. 5, p. 447, 2010.
10. Chidamber, S.R. and Kemerer, C.F., 1991. *Towards a metrics suite for object oriented design* (Vol. 26, No. 11, pp. 197-211). ACM.
11. Subramanyam, R. and Krishnan, M.S., 2003. Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects. *IEEE Transactions on software engineering*, *29*(4), pp.297-310.
12. Zhao, J., 2004, September. Measuring coupling in aspect-oriented systems. In *10th International Software Metrics Symposium (Metrics 04)*.
13. Rothenberger, M.A., Dooley, K.J., Kulkarni, U.R. and Nada, N., 2003. Strategies for software reuse: A principal component analysis of reuse practices. *IEEE Transactions on Software Engineering*, *29*(9), pp.825-837.
14. Mens, T. and Tourwé, T., 2004. A survey of software refactoring. *IEEE Transactions on software engineering*, *30*(2), pp.126-139.
15. Singh, S. and Kahlon, K.S., 2011. Effectiveness of encapsulation and object-oriented metrics to refactor code and identify error prone classes using bad smells. *ACM SIGSOFT Software Engineering Notes*, *36*(5), pp.1-10.
16. Burrows, R., Ferrari, F.C., Garcia, A. and Taïani, F., 2010, May. An empirical evaluation of coupling metrics on aspect-oriented programs. In *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics* (pp. 53-58). ACM.
17. Dhambri, K., Gélinas, J.F., Hassaine, S. and Langelier, G., Visualization-based Analysis of Quality for Aspect-Oriented Systems.
18. Mguni, K. and Ayalew, Y., 2013. An Assessment of Maintainability of an Aspect-Oriented System. ISRN Software Engineering, 2013.
19. Piveta, E.K., Moreira, A., Pimenta, M.S., Araújo, J., Guerreiro, P. and Price, R.T., 2012. An empirical study of aspect-oriented metrics. Science of Computer Programming, 78(1), pp.117-144.
20. Gélinas, J.F., Badri, M. and Badri, L., 2006. A Cohesion Measure for Aspects. *Journal of object technology*, *5*(7), pp.75-95.
21. Arora, K., Singhal, A. and Kumar, A., 2012. A Study of Cohesion Metrics for Aspect Oriented Systems. *Int. J. Eng. Sci. Adv. Tech*, *2*(2), pp.332-337.
22. Bartolomei, T.T., Garcia, A., Sant'Anna, C. and Figueiredo, E., 2006, November. Towards a unified coupling framework for measuring aspect-oriented programs. In *Proceedings of the 3rd international workshop on Software quality assurance* (pp. 46-53). ACM.
23. Burrows, R., Ferrari, F.C., Garcia, A. and Taïani, F., 2010, May. An empirical evaluation of coupling metrics on aspect-oriented programs. In *Proceedings of the 2010 ICSE Workshop on Emerging Trends in Software Metrics* (pp. 53-58). ACM.
24. Ghareb, M.I. and Allen, G., 2015. Identifying Similar Pattern of Potential Aspect Oriented Functionalities in Software Development Life Cycle. *Journal of Theoretical and Applied Information Technology*, *80*(3), p.491.

25. Ghareb, M. and Allen, G., 2015. Improving the Design and Implementation of Software Systems uses Aspect Oriented Programming.
26. Burrows, R., Garcia, A. and Taïani, F., 2008, May. Coupling metrics for aspect-oriented programming: A systematic review of maintainability studies. In *International Conference on Evaluation of Novel Approaches to Software Engineering* (pp. 277-290). Springer, Berlin, Heidelberg.
27. Sirbi, K. and Kulkarni, P.J., 2010. Impact of aspect-oriented programming on software development design quality metrics-A comparative study. *Journal of Computing*, *2*(7).
28. Pataki, N., Sipos, A. and Porkolab, Z., 2006, July. Measuring the complexity of aspect-oriented programs with multiparadigm metric. In *Proceedings of the 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2006)* (pp. 1-10).
29. Debnath, N., Baigorria, L., Riesco, D. and Montejano, G., 2008, July. Metrics applied to Aspect Oriented Design using UML profiles. In *Computers and Communications, 2008. ISCC 2008. IEEE Symposium on* (pp. 654-657). IEEE.
30. Babu, C. and Vijayalakshmi, R., 2008. Metrics-based design selection tool for aspect oriented software development. *ACM SIGSOFT Software Engineering Notes*, *33*(5), p.4.
31. Sirbi, K. and Kulkarni, P.J., 2010. Metrics for Aspect Oriented Programming-An Empirical Study. *International Journal of Computer Applications (0975–8887)*, *5*(12).
32. Sant'Anna, C., Garcia, A., Chavez, C., Lucena, C. and Von Staa, A., 2003, October. On the reuse and maintenance of aspect-oriented software: An assessment framework. In *Proceedings of Brazilian symposium on software engineering* (pp. 19-34).
33. Mehner, K., 2006. On Using Metrics in the Evaluation of Aspect-Oriented Programs and Designs. In *LATE Workshop AOSD*.
34. Vinobha, A., Velan, S. and Babu, C., 2014, May. Evaluation of reusability in Aspect Oriented Software using inheritance metrics. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on* (pp. 1715-1722). IEEE.
35. Vinobha, A., Velan, S. and Babu, C., 2014, May. Evaluation of reusability in Aspect Oriented Software using inheritance metrics. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on* (pp. 1715-1722). IEEE.
36. Sirbi, K. and Kulkarni, P.J., 2013. AOP and its impact on software quality. *Elixir Computer Science and Engineering*, pp.12606-12610.
37. Rønningen, E. and Steinmoen, T., 2004. *Metrics for Aspect-Oriented Programming of middleware systems* (Master's thesis, Institutt for datateknikk og informasjonsvitenskap).
38. Przybylek, A., 2010, July. An Empirical Assessment of the Impact of Aspect-oriented Programming on Software Modularity. In *ENASE* (pp. 139-148).
39. Tsang, S.L., Clarke, S. and Baniassad, E., 2004. Object metrics for aspect systems: Limiting empirical inference based on modularity. *Submitted to ECOOP*.
40. Greenwood, P., Bartolomei, T., Figueiredo, E., Dosea, M., Garcia, A., Cacho, N., Sant'Anna, C., Soares, S., Borba, P., Kulesza, U. and Rashid, A., 2007, July. On the impact of aspectual decompositions on design stability: An empirical study. In *European Conference on Object-Oriented Programming*(pp. 176-200). Springer Berlin Heidelberg.
41. Zhang, C., 2000. Hans-Arno. Jacobsen. Quantifying Aspects in Middleware Platforms. Department of Electrical and Computer Engineering and Department of Computer Science, University of Toronto.
42. Coady, Y. and Kiczales, G., 2003, March. Back to the future: a retroactive study of aspect evolution in operating system code. In *Proceedings of the 2nd international conference on Aspect-oriented software development* (pp. 50-59). ACM.
43. Zakaria, A.A. and Hosny, H., 2003, March. Metrics for aspect-oriented software design. In *Proc. Third International Workshop on Aspect-Oriented Modeling, AOSD* (Vol. 3).
44. Zhao, J., 2002. Towards a metrics suite for aspect-oriented software. *Rapport technique*.
45. Dospisil, J. and Khemngoen, A., 2003, June. Measuring the Complexity of Mobile Agents Designed with Aspect/J. In *Informing Science+ IT Education (InSITE) Conference, Finland*.
46. Dospisil, J., 2003. Measuring Code Complexity in Projects Designed with AspectJ. *Informing Science InSITE-"Where Parallels Intersects*, *23*, pp.185-196.
47. E. Kirubakaran, E., & Martin, K. R. (2016). A Review on Coupling Metrics in Aspect Oriented System. International Journal of Control Theory and Applications, (9(27), 2016, pp. 93-97). Retrieved from https:www.researchgate.net

48. Alemneh, E., 2014. Current States of Aspect Oriented Programming Metrics. *International Journal of Science and Research*, *3*(1), pp.142-146.
49. Madeyski, L., 2007. Impact of aspect-oriented programming on software development efficiency and design quality: an empirical study. *Iet Software*, *1*(5), pp.180-187.
50. Sirbi, K. and Kulkarni, P.J., 2010. Impact of Aspect Oriented Programming on Software Development Quality Metrics. *Global Journal of Computer Science and Technology*.
51. Pataki, N., Sipos, A. and Porkolab, Z., 2006, July. Measuring the complexity of aspect-oriented programs with multiparadigm metric. In *Proceedings of the 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2006)* (pp. 1-10).
52. Hannemann, J. and Kiczales, G., 2002, November. Design pattern implementation in Java and AspectJ. In *ACM Sigplan Notices* (Vol. 37, No. 11, pp. 161-173). ACM.
53. Lopez-Herrejon, R.E. and Apel, S., 2007, March. Measuring and characterizing crosscutting in aspect-based programs: Basic metrics and case studies. In *FASE* (Vol. 7, pp. 423-437).
54. Mehner, K., 2006. On Using Metrics in the Evaluation of Aspect-Oriented Programs and Designs. In *LATE Workshop AOSD*.
55. Gélinas, J.F., Badri, M. and Badri, L., 2006. A Cohesion Measure for Aspects. *Journal of object technology*, *5*(7), pp.75-95.
56. Kaur, M. and Kaur, R., 2015. Improving the design of Cohesion and coupling metrics for aspect oriented software development. *Int. J. Comput. Sci. Mob. Comput*, *4*(5), pp.99-106.
57. Kaur, Mandeep, and Rupinder Kaur. "Improving the design of Cohesion and coupling metrics for aspect oriented software development." *Int. J. Comput. Sci. Mob. Comput* 4, no. 5 (2015): 99-106.
58. Sharma, D. and Narula, P., 2014. Static and Dynamic Analysis of Object Oriented Systems. *International Journal*, *4*(6).
59. Zavvar, M., Garavand, S., Nehi, M.R., Yanpi, A., Rezaei, M. and Zavvar, M.H., 2017. Measuring Reliability of Aspect-Oriented Software Using a Combination of Artificial Neural Network and Imperialist Competitive Algorithm. *Asia-Pacific Journal of Information Technology and Multimedia*, *5*(2).
60. Cacho, N., Sant'Anna, C., Figueiredo, E., Garcia, A., Batista, T. and Lucena, C., 2006, March. Composing design patterns: a scalability study of aspect-oriented programming. In *Proceedings of the 5th international conference on Aspect-oriented software development* (pp. 109-121). ACM.
61. Zavvar, M., Garavand, S., Nehi, M.R., Yanpi, A., Rezaei, M. and Zavvar, M.H., 2017. Measuring Reliability of Aspect-Oriented Software Using a Combination of Artificial Neural Network and Imperialist Competitive Algorithm. *Asia-Pacific Journal of Information Technology and Multimedia*, *5*(2).
62. Garcia, A., Sant'Anna, C., Figueiredo, E., Kulesza, U., Lucena, C. and von Staa, A., 2006. Modularizing design patterns with aspects: a quantitative study. In *Transactions on Aspect-Oriented Software Development I* (pp. 36-74). Springer, Berlin, Heidelberg.
63. Kersten, M. and Murphy, G.C., 1999, October. Atlas: a case study in building a web-based learning environment using aspect-oriented programming. In *ACM SIGPLAN Notices* (Vol. 34, No. 10, pp. 340-352). ACM.
64. Sant'Anna, C., Garcia, A., Kulesza, U., Lucena, C. and Staa, A.V., 2004. Design patterns as aspects: A quantitative assessment. *Journal of the Brazilian Computer Society*, *10*(2), pp.42-55.
65. Alencar, P., Barrenechea, E., Garcia, A., Lucena, C. and Cowan, D., 2004. *A Query-Based Approach for Aspect Measurement and Analysis*. TR CS-2004-13, School of Computer Science, Univ. of Waterloo, Canada.
66. Garcia, A., Sant'Anna, C., Chavez, C., da Silva, V.T., de Lucena, C.J. and von Staa, A., 2003, May. Separation of concerns in multi-agent systems: An empirical study. In *International Workshop on Software Engineering for Large-Scale Multi-agent Systems* (pp. 49-72). Springer, Berlin, Heidelberg.
67. M. P. Brown and K. Austin, *The New Physique* (Publisher Name, Publisher City, 2005), pp. 25–30.
68. M. P. Brown and K. Austin, Appl. Phys. Letters **85**, 2503–2504 (2004).
69. R. T. Wang, "Title of Chapter," in *Classic Physiques*, edited by R. B. Hamil (Publisher Name, Publisher City, 1999), pp. 212–213.
70. C. D. Smith and E. F. Jones, "Load-cycling in cubic press," in *Shock Compression of Condensed Matter-2001*, AIP Conference Proceedings 620, edited by M. D. Furnish *et al*. (American Institute of Physics, Melville, NY, 2002), pp. 651–654.
71. B. R. Jackson and T. Pitman, U.S. Patent No. 6,345,224 (8 July 2004)
72. D. L. Davids, "Recovery effects in binary aluminum alloys," Ph.D. thesis, Harvard University, 1998.
73. R. C. Mikkelson (private communication).